Control 7 segment digits by keyboard

In the previous tutorial, we learn about uart data transmit. In this tutorial, we will start uart receive data from the PC terminal to the STM32 dev board and Control 7-segment display digit by simply typing the keyboard digit.

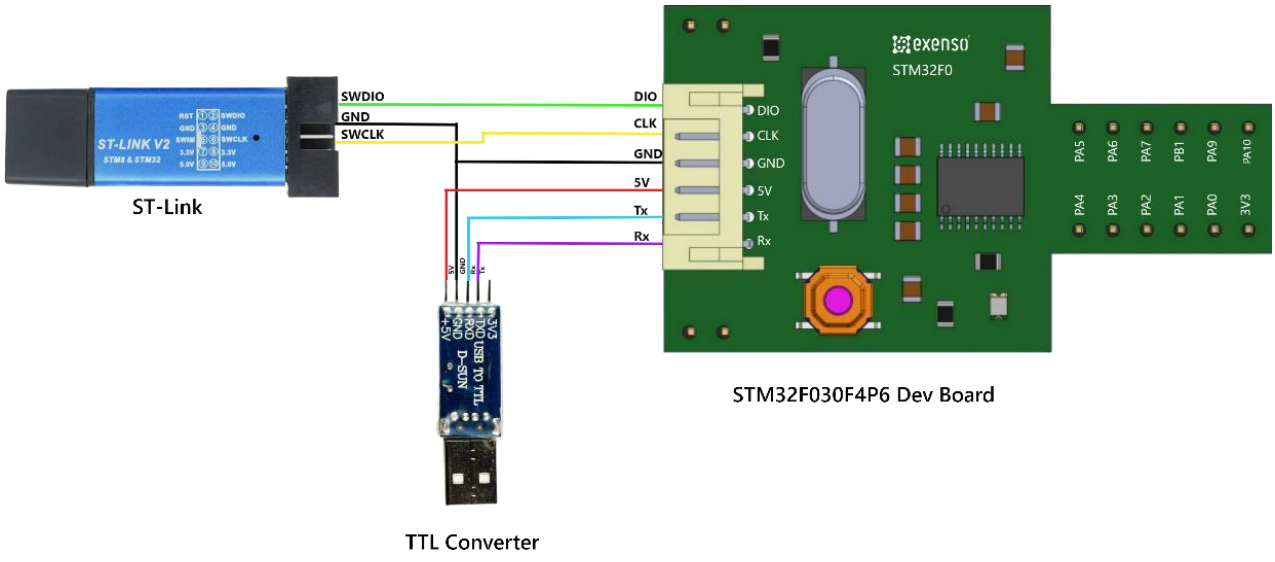
Components Required

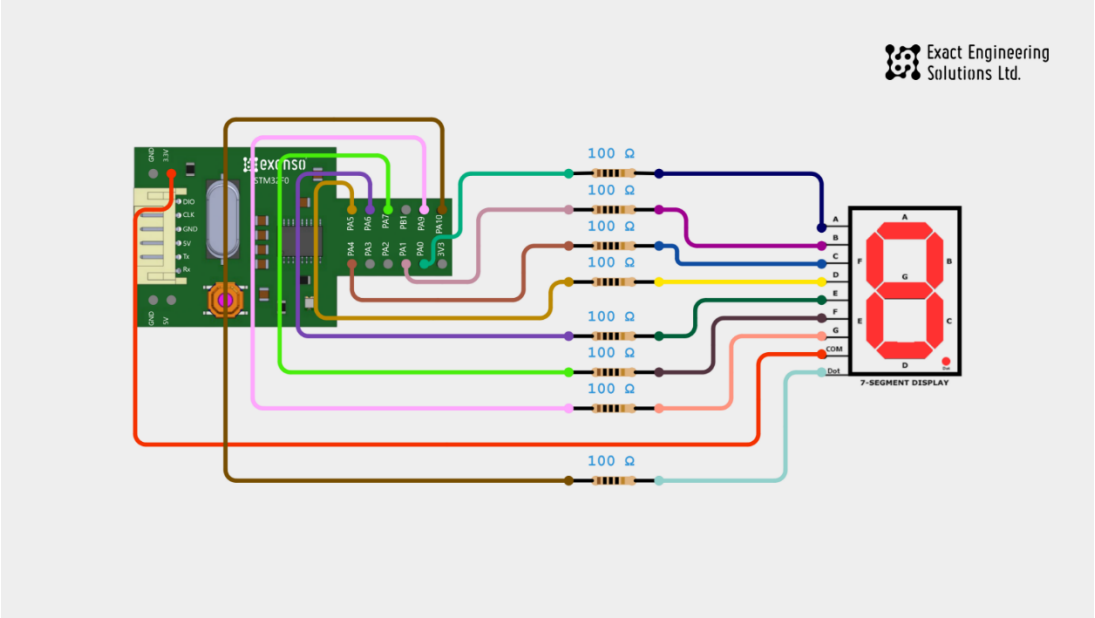
You will need the following components −

- 1 × Breadboard
- 1 × STM32F030F4P6
- 1× TTL Converter
- 1× Single digit Seven Segment display
- 8 × 100Ω Resistor
- Some Jumper wire

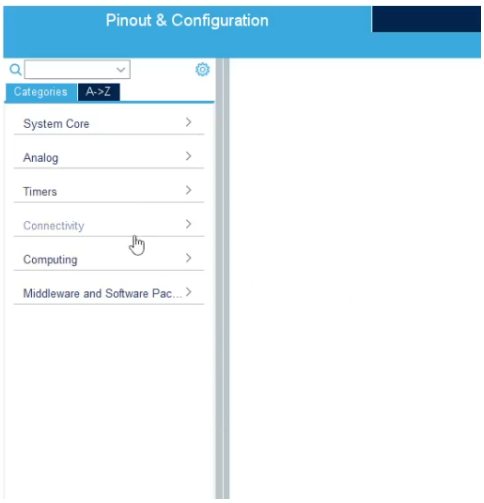
Procedure

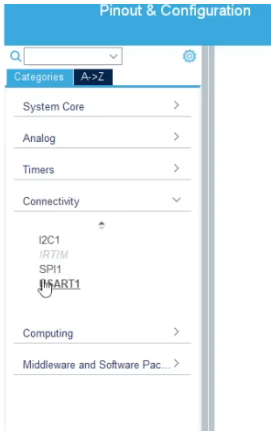
Follow the circuit diagram shown in the image given below.

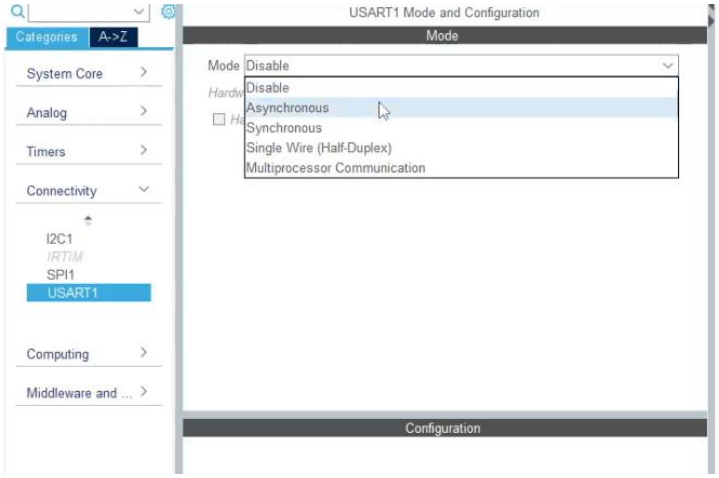




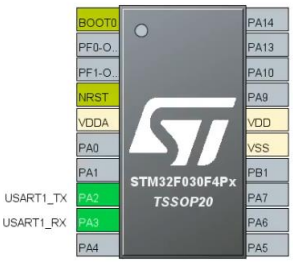
STM32F0 Pin Configuration:



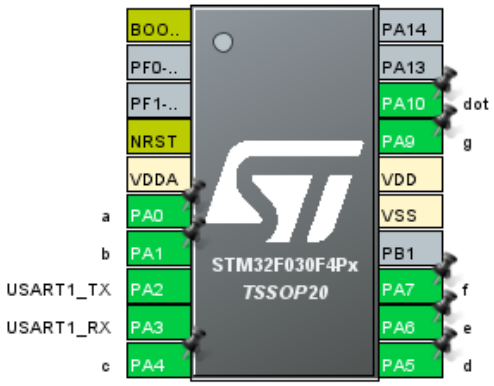



USART1 Mode and Configuration
Mode
Categories
A->Z
System Core
Analog
Timers
Connectivity
I2C1
SPI1
USART1
Computing
Middleware and ...
Mode
Disable
Disable
Asynchronous
Synchronous
Single Wire (Half-Duplex)
Multiprocessor Communication
Configuration


Mode
Asynchronous
Hardware Flow Control (RS232)
Disable
Hardware Flow Control (RS485)
Configuration
Reset Configuration
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
Configure the below parameters :
Search (Ctrl+F)
Basic Parameters
Baud Rate
38400
Word Length
8 Bits (including Parity)
Parity
None
Stop Bits
1
Advanced Parameters
Data Direction
Receive and Transmit


BOOT0
PF0-O..
PF1-O..
NRST
VDDA
PA0
PA1
PA2
PA3
PA4
PA14
PA13
PA10
PA9
VDD
VSS
PB1
PA7
PA6
PA5
USART1_TX
USART1_RX
STM32F030F4Px
TSSOP20

PA0 to PA1 and PA4 to PA10 set as OUTPUT



Code

```
#include "main.h"

UART_HandleTypeDef huart1;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART1_UART_Init();
  uint8_t UART1_rxBuffer[2] = {0};
  while (1)
  {

        HAL_UART_Receive(&huart1, UART1_rxBuffer, 2, 100);

        switch (UART1_rxBuffer[0]){
            case '0':
                                // statements
                               //0
                                  HAL_GPIO_WritePin(GPIOA, a_Pin, GPIO_PIN_RESET);
                                  HAL_GPIO_WritePin(GPIOA, b_Pin, GPIO_PIN_RESET);
                                  HAL_GPIO_WritePin(GPIOA, c_Pin, GPIO_PIN_RESET);
                                  HAL_GPIO_WritePin(GPIOA, d_Pin, GPIO_PIN_RESET);
                                  HAL_GPIO_WritePin(GPIOA, e_Pin, GPIO_PIN_RESET);
                                  HAL_GPIO_WritePin(GPIOA, f_Pin, GPIO_PIN_RESET);
                                  HAL_GPIO_WritePin(GPIOA, g_Pin, GPIO_PIN_SET);
                                  HAL_GPIO_WritePin(GPIOA, dot_Pin, GPIO_PIN_SET);
                        break;

            case '1':
                        // statements
                                  //1
                                  HAL_GPIO_WritePin(GPIOA, a_Pin, GPIO_PIN_SET);
                                  HAL_GPIO_WritePin(GPIOA, b_Pin, GPIO_PIN_RESET);
                                  HAL_GPIO_WritePin(GPIOA, c_Pin, GPIO_PIN_RESET);
                                  HAL_GPIO_WritePin(GPIOA, d_Pin, GPIO_PIN_SET);
                                  HAL_GPIO_WritePin(GPIOA, e_Pin, GPIO_PIN_SET);
                                  HAL_GPIO_WritePin(GPIOA, f_Pin, GPIO_PIN_SET);
                                  HAL_GPIO_WritePin(GPIOA, g_Pin, GPIO_PIN_SET);
                                  HAL_GPIO_WritePin(GPIOA, dot_Pin, GPIO_PIN_SET);

                        break;
            case '2':
                        // statements
                                  //2
                                  HAL_GPIO_WritePin(GPIOA, a_Pin, GPIO_PIN_RESET);
                                  HAL_GPIO_WritePin(GPIOA, b_Pin, GPIO_PIN_RESET);
                                  HAL_GPIO_WritePin(GPIOA, c_Pin, GPIO_PIN_SET);
                                  HAL_GPIO_WritePin(GPIOA, d_Pin, GPIO_PIN_RESET);
                                  HAL_GPIO_WritePin(GPIOA, e_Pin, GPIO_PIN_RESET);
                                  HAL_GPIO_WritePin(GPIOA, f_Pin, GPIO_PIN_SET);
                                  HAL_GPIO_WritePin(GPIOA, g_Pin, GPIO_PIN_RESET);
                                  HAL_GPIO_WritePin(GPIOA, dot_Pin, GPIO_PIN_SET);
                        break;
```

```
        case '3':
                // statements
                        //3
                        HAL_GPIO_WritePin(GPIOA, a_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, b_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, c_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, d_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, e_Pin, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOA, f_Pin, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOA, g_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, dot_Pin, GPIO_PIN_SET);
                break;
        case '4':
                // statements
                        //4
                        HAL_GPIO_WritePin(GPIOA, a_Pin, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOA, b_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, c_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, d_Pin, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOA, e_Pin, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOA, f_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, g_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, dot_Pin, GPIO_PIN_SET);
                break;
        case '5':
                // statements
                        //5
                        HAL_GPIO_WritePin(GPIOA, a_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, b_Pin, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOA, c_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, d_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, e_Pin, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOA, f_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, g_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, dot_Pin, GPIO_PIN_SET);
                break;
        case '6':
                // statements
                        //6
                        HAL_GPIO_WritePin(GPIOA, a_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, b_Pin, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOA, c_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, d_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, e_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, f_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, g_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, dot_Pin, GPIO_PIN_SET);
                break;
        case '7':
                // statements
                        //7
                        HAL_GPIO_WritePin(GPIOA, a_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, b_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, c_Pin, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOA, d_Pin, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOA, e_Pin, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOA, f_Pin, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOA, g_Pin, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOA, dot_Pin, GPIO_PIN_SET);
                break;
```

```
        case '8':
                  // statements
                                //8
                                HAL_GPIO_WritePin(GPIOA, a_Pin, GPIO_PIN_RESET);
                                HAL_GPIO_WritePin(GPIOA, b_Pin, GPIO_PIN_RESET);
                                HAL_GPIO_WritePin(GPIOA, c_Pin, GPIO_PIN_RESET);
                                HAL_GPIO_WritePin(GPIOA, d_Pin, GPIO_PIN_RESET);
                                HAL_GPIO_WritePin(GPIOA, e_Pin, GPIO_PIN_RESET);
                                HAL_GPIO_WritePin(GPIOA, f_Pin, GPIO_PIN_RESET);
                                HAL_GPIO_WritePin(GPIOA, g_Pin, GPIO_PIN_RESET);
                                HAL_GPIO_WritePin(GPIOA, dot_Pin, GPIO_PIN_SET);
                  break;
         case '9':
                 // statements
                                //9
                                HAL_GPIO_WritePin(GPIOA, a_Pin, GPIO_PIN_RESET);
                                HAL_GPIO_WritePin(GPIOA, b_Pin, GPIO_PIN_RESET);
                                HAL_GPIO_WritePin(GPIOA, c_Pin, GPIO_PIN_RESET);
                                HAL_GPIO_WritePin(GPIOA, d_Pin, GPIO_PIN_RESET);
                                HAL_GPIO_WritePin(GPIOA, e_Pin, GPIO_PIN_SET);
                                HAL_GPIO_WritePin(GPIOA, f_Pin, GPIO_PIN_RESET);
                                HAL_GPIO_WritePin(GPIOA, g_Pin, GPIO_PIN_RESET);
                                HAL_GPIO_WritePin(GPIOA, dot_Pin, GPIO_PIN_SET);
                  break;

         default:

                                HAL_GPIO_WritePin(GPIOA, a_Pin, GPIO_PIN_SET);
                                HAL_GPIO_WritePin(GPIOA, b_Pin, GPIO_PIN_SET);
                                HAL_GPIO_WritePin(GPIOA, c_Pin, GPIO_PIN_SET);
                                HAL_GPIO_WritePin(GPIOA, d_Pin, GPIO_PIN_SET);
                                HAL_GPIO_WritePin(GPIOA, e_Pin, GPIO_PIN_SET);
                                HAL_GPIO_WritePin(GPIOA, f_Pin, GPIO_PIN_SET);
                                HAL_GPIO_WritePin(GPIOA, g_Pin, GPIO_PIN_SET);
           // default statements
       }

  }

}
```

We can try another way. GPIO remapping add LIB “GPIO_Remap.h”

Code

```
#include "main.h"

#include "GPIO_Remap.h"

UART_HandleTypeDef huart1;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);

uint8_t seg_table[] = {
    //pgfedcba
    0b11000000,  // 0
    0b11111001,  // 1
    0b10100100,  // 2
    0b10110000,  // 3
    0b10011001,  // 4
    0b10010010,  // 5
    0b10000010,  // 6
    0b11111000,  // 7
    0b10000000,  // 8
    0b10010000   // 9
};

PORT_Remap_TypeDef PORTrm;
uint8_t UART1_rxBuffer[2] = {0};
uint8_t ch0;

int main(void)
{

  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART1_UART_Init();

 // Remaping GPIO PORT AND PIN

  PORTrm.PORT[0] = GPIOA; PORTrm.PIN[0] = a_Pin;
  PORTrm.PORT[1] = GPIOA; PORTrm.PIN[1] = b_Pin;
  PORTrm.PORT[2] = GPIOA; PORTrm.PIN[2] = c_Pin;
  PORTrm.PORT[3] = GPIOA; PORTrm.PIN[3] = d_Pin;
  PORTrm.PORT[4] = GPIOA; PORTrm.PIN[4] = e_Pin;
  PORTrm.PORT[5] = GPIOA; PORTrm.PIN[5] = f_Pin;
  PORTrm.PORT[6] = GPIOA; PORTrm.PIN[6] = g_Pin;
  PORTrm.PORT[7] = GPIOA; PORTrm.PIN[7] = dot_Pin;

  while (1)
  {

        HAL_UART_Receive (&huart1, UART1_rxBuffer, 2, 100);

        switch (UART1_rxBuffer[0]){
            case '0':
                                // statements
                        //0
                    Write_PORT_Remap(&PORTrm, seg_table[0]);
                      break;
```

```
            case '1':
                    // statements
                                //1
                    Write_PORT_Remap(&PORTrm, seg_table[1]);
                    break;



            case '2':
                    // statements
                                //2
                   Write_PORT_Remap(&PORTrm, seg_table[2]);
                    break;
            case '3':
                    // statements
                                //3
                    Write_PORT_Remap(&PORTrm, seg_table[3]);
                    break;
            case '4':
                    // statements
                                //4
                    Write_PORT_Remap(&PORTrm, seg_table[4]);
                    break;
            case '5':
                    // statements
                                //5
                         Write_PORT_Remap(&PORTrm, seg_table[5]);
                    break;
            case '6':
                    // statements
                                //6
                         Write_PORT_Remap(&PORTrm, seg_table[6]);
                    break;
            case '7':
                    // statements
                                //7
                    Write_PORT_Remap(&PORTrm, seg_table[7]);
                    break;
            case '8':
                    // statements
                                //8
                    Write_PORT_Remap(&PORTrm, seg_table[8]);
                    break;
            case '9':
                   // statements
                                //9
                    Write_PORT_Remap(&PORTrm, seg_table[9]);
                    break;

            default:
                   Write_PORT_Remap(&PORTrm, seg_table[0]);
              // default statements
        }
      }
  }
```

```
HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t
Timeout)
```